TPs PIC / SOLAIRE

2ème année L3 EEA à distance

Année 2011/2012

Etudiant : Yann Le Fustec

I Prise en main du système

Dans un premier temps il a fallu câbler le PIC pour le programmer et tester le premier programme 1st_Project_pic_c. :

```
void main() {
  PORTC = 0;              // Initialize PORTC
  TRISC = 0;              // Configure PORTC as output

  while(1) {
    PORTC =  ~PORTC;      // toggle PORTC
    Delay_ms(1000);       // one second delay
  }
}
```

Ce programme fait clignoter une LED connectée sur une broche du PIC (RC2) définie en sortie. La fonction Delay_ms() permet de faire varier la fréquence de clignotement. La valeur en argument correspond à une demi-période. Dans l'exemple toutes les sorties du port C alternent.

I.1 *Cablâge*

Pin diagram PIC16F877 :

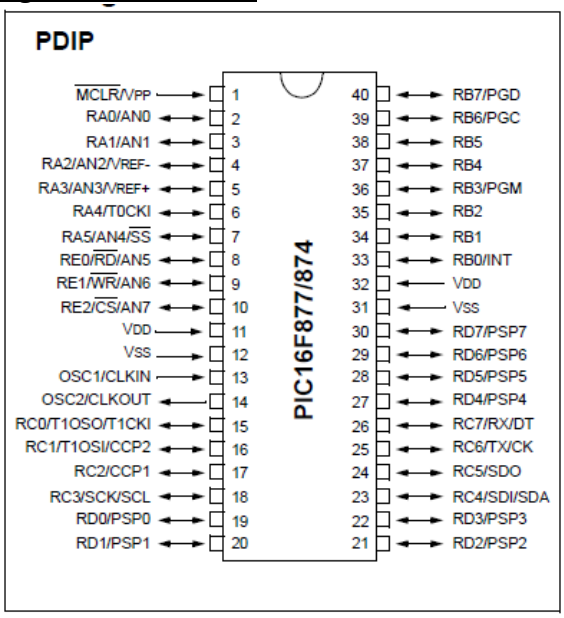
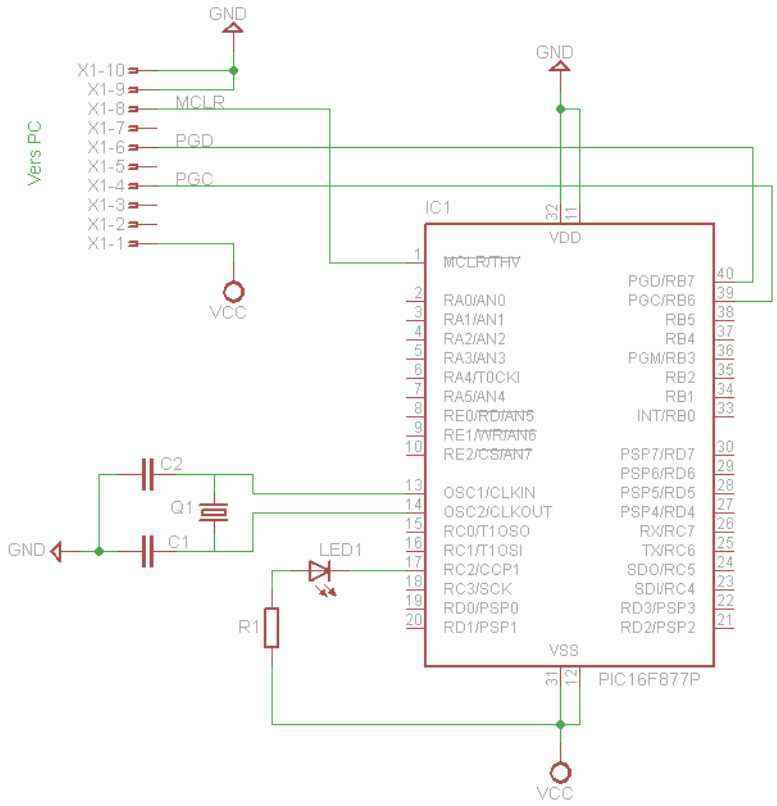


Schéma du câblage :



II Utilisation d'un afficheur LCD

Dans un premier temps j'ai testé l'afficheur avec le programme suivant :

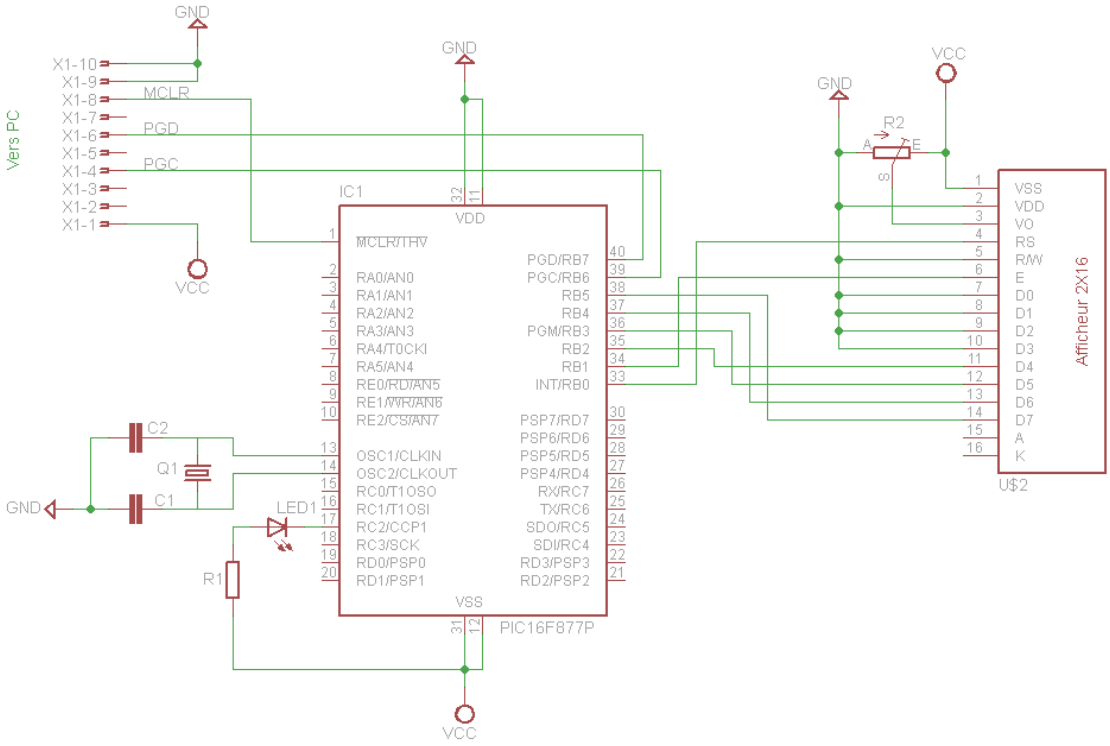
```
void main()
{
  PORTC = 0;              // Initialize PORTC
  TRISC = 0;              // Configure PORTC as output
  TRISB = 0;              // PORTB is output

  Lcd_Config(&PORTB, 0,1,WR,5,4,3,2);        // Initialize LCD
  Lcd_Cmd(Lcd_CLEAR);      // Clear display
  Lcd_Cmd(Lcd_CURSOR_OFF);  // Turn cursor off
  while(1)
  {
    Lcd_Out(1, 1, "Hello world");      // Print text to LCD, 1st row, 1st column
    PORTC =  ~PORTC;        // toggle PORTC
    Delay_ms(1000);         // one second delay

  }
}
```

Ce programme affiche le texte sur la première ligne du LCD et fait clignoter la LED comme précédemment.

Configuration du LCD :

- PORT B en sortie
- Fonction Lcd_Config - définition du PORT et assignement des signaux RS, EN, WR et 4 bits de data : void Lcd_Config(char *port, char RS, char EN, char WR, char D7, char D6, char D5, char D4)

II.1 *Câblage*



III Utilisation d'un convertisseur analogique numérique pour contrôler une sortie MLI

Dans cette partie j'ai configuré le PIC pour utiliser une des ses entrées CAN (A0) connectée à un potentiomètre qui fait varier la tension de 0 à 5V (VSS). Le programme permet de lire la valeur de tension sur A0 et de calculer un rapport cyclique α en fonction de cette valeur pour contrôler une sortie MLI. Les valeurs de α et de la tension sont affichées sur le LCD.

Puisque la sortie MLI servira par la suite à commandé un hacheur, j'ai ajouté au programme des butées à 10% et 90% de rapport cyclique.

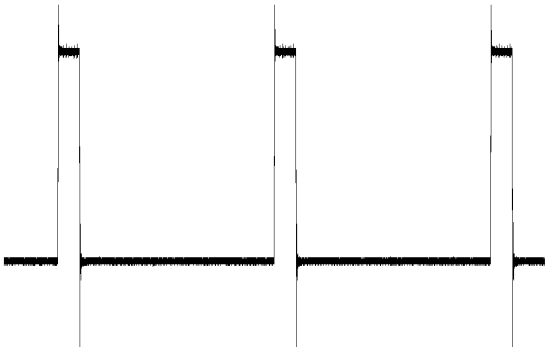
Le programme est le suivant :

```
unsigned int ValeurAD;
unsigned int duty_ratio;
int LowLimit;
int HighLimit;
char Alpha[7];
char TENSION[7];
char strValAD[7];
```

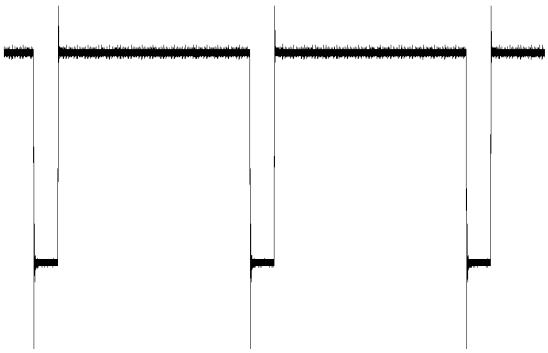
```
void main()
{
  //initalisation des butées
  LowLimit = 26; //10% de 255
  HighLimit = 230; //90% de 255

  //TRISB = 0xC0; // PORTB is output

  PORTC = 0;              // Initialise PORTC
  TRISC = 0;              // Configure PORTC en sortie
  TRISB = 0;              // Configure PORTB en sortie
  ADCON1 = 0x8E;      // A0 entrée analogique
  TRISA = 1;              // Configure PORTA en entrée

   //initialisation port LCD
  Lcd_Config(&PORTB, 0,1,WR,5,4,3,2);
  Lcd_Cmd(Lcd_CURSOR_OFF);

  //initialisation de la fréquence de la PWM
  Pwm_Init(50000);   //50k
  Pwm_Change_Duty(0); //rapport cyclique initialise à 0

  //démarrage MLI
  Pwm_Start();

  while(1)
  {
    Lcd_Cmd(Lcd_CLEAR);

    //lecture de la valeur
    ValeurAD = Adc_Read(0);
    IntToStr(ValeurAD,strValAD);    //converti en 'string' pour écriture lcd
    IntToStr(((ValeurAD*5)/1.024), TENSION);

    //calcul de alpha  (fonction de la tension)
    duty_ratio = ((ValeurAD*2.55)/10.24);
    if (duty_ratio < LowLimit)
    {
      duty_ratio = LowLimit;
    }
    if (duty_ratio > HighLimit)
    {
      duty_ratio = HighLimit;
    }
    IntToStr(duty_ratio,Alpha);

    //affichage
    Lcd_Out(1, 1, "Alpha=");
    Lcd_Out(1, 7, Alpha);
    Lcd_Out(2, 1, "TENSION=");
    Lcd_Out(2, 9, TENSION);

    Pwm_Change_Duty(duty_ratio);

    Delay_ms(10);          // delay
  }
}
```

Le programme a été testé (affichage) et j'ai bien vérifié le fonctionnement des butées en relevant la sortie MLI sur l'oscilloscope :

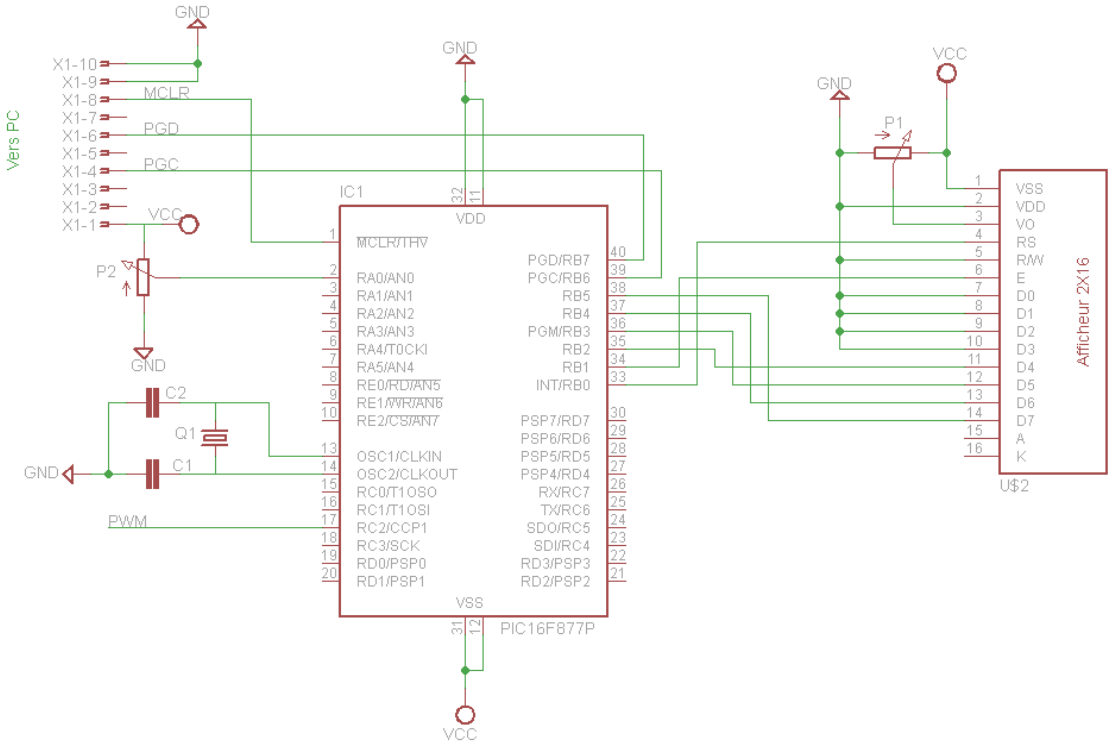
A 10% :



Et 90% :



III.1 *Câblage*



IV Application à un panneau photovoltaïque

IV.1 *Caractérisation du panneau*

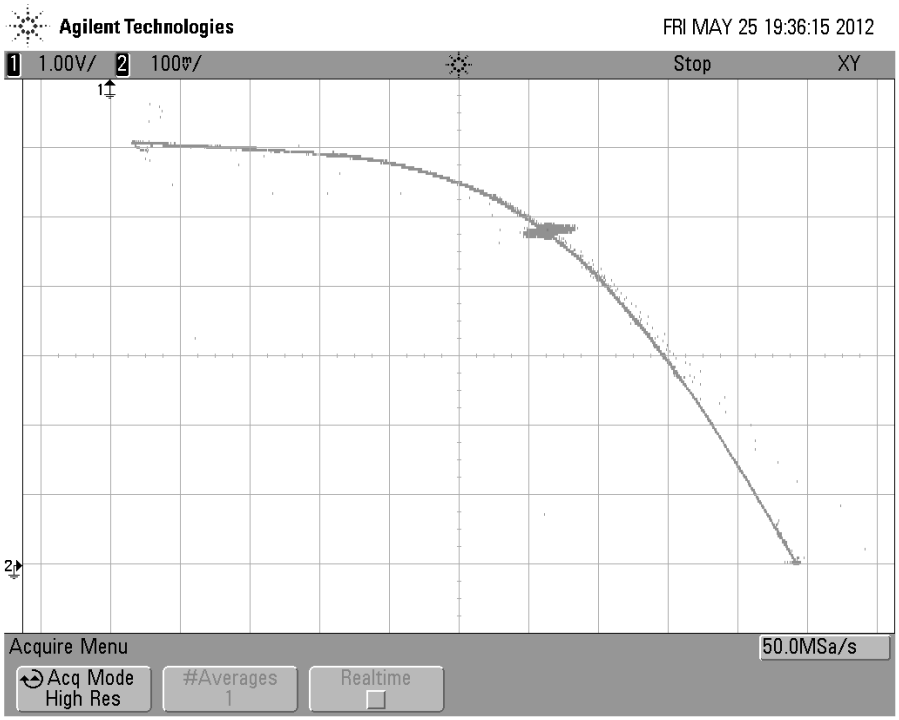
Cette partie a été réalisée durant la session de projet sur l'étude d'un module PV de type polycristallin. Les caractéristiques du panneau sont les suivantes :

CARACTERISTIQUES ELECTRIQUES							
PW1650		**Configuration 24 V**			**Configuration 12 V**		
Puissance typique	**W**	155	165	175	155	165	175
Puissance minimale	**W**	150	160	170	150	160	170
Tension à la puissance typique	**V**	34	34,4	35	17	17,2	17,5
Intensité à la puissance typique	**A**	4,6	4,8	5,0	9,2	9,6	10
Intensité de court circuit	**A**	4,8	5,1	5,3	9,6	10,2	10,6
Tension en circuit ouvert	**V**	43	43,2	43,4	21,5	21,6	21,7
Tension maximum du circuit	**V**	**770V DC**					
Coefficient de température		α=+1,46 mA/°C ; β=-158 mV/°C ; γ P/ P=-0,43 %/°C			α = +2,92 mA/°C ; β = -79 mV/°C ; γ P/P = - 0,43 %/°C		
Specifications de puissance à 1000 W/m² : 25°C : AM 1,5							

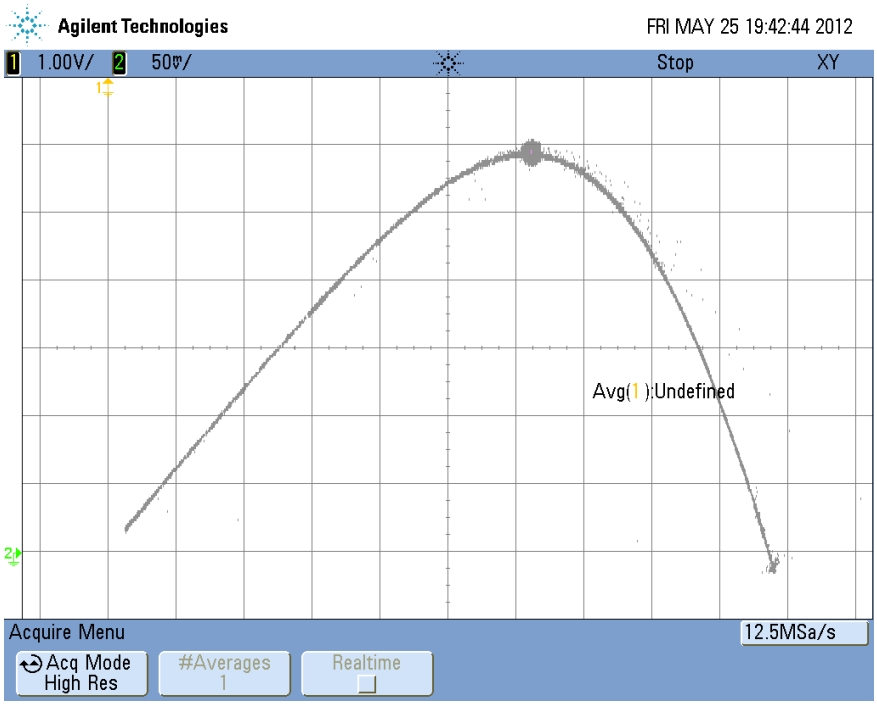
Les caractéristiques de I(V) et de P(V) ont été relevées pour une irradiation constante mesurée grâce à un pyranomètre.

Irradiation : $8.76 \times 11 \approx 972\ W.m^{-2}$

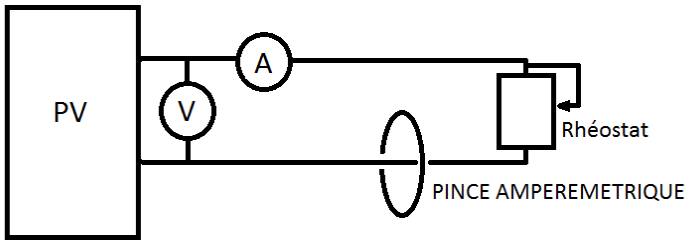
I(V) :



P(V) :



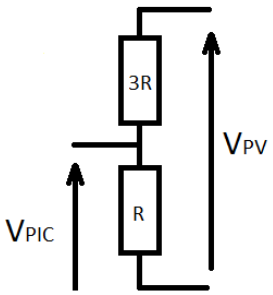
Les points sur les deux courbes représentent le point de fonctionnement pour obtenir le maximum de puissance.
Les courants de court-circuit et tension de circuit ouvert ont été mesurées grâce au montage suivant :



Le rhéostat était en fait constitué de deux rhéostats de 30Ohms et 1000Ohms pour pouvoir faire varier la caractéristique du courant de court-circuit jusqu'à la tension de circuit ouvert. Les valeurs mesurées sont les suivantes :
Vco = 20V
Icc = 6.15A
La pince ampèremétrique a permit de sortir la mesure de courant sur un oscilloscope et sur un boitier 'multiplieur' dont la deuxième entrée recevait la mesure de tension pour obtenir ainsi une mesure de puissance.

IV.2 *Mesure des courant et tension pour exploitation par le PIC*

La plage de tension du PV allant à peu près de 0 à 20V (21,7V d'après la caractéristique constructeur), pour pouvoir l'exploiter sur le PIC il a fallu la ramener à une plage allant de 0 à 5V. Pour cela un simple diviseur de tension fait l'affaire :

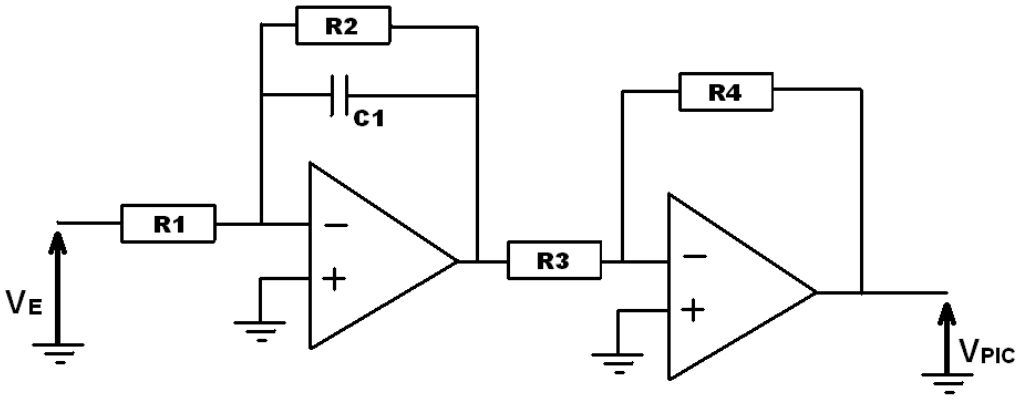


Il faut utiliser des valeurs suffisantes pour limiter les pertes dues aux mesures – de l'ordre du K Ω. Les résistances utilisées ont été mesurées à l'ohmètre pour s'approcher le plus possible de la théorie.

- ⇨ 3R = 63KΩ
- ⇨ R = 21KΩ

Le bon fonctionnement a pu être vérifié sur le montage : pour une tension PV de 19,98V, la tension mesurée au voltmètre après le pont diviseur est de 4,93V, et celle lue sur l'afficheur est de 4,86V. On obtient donc une erreur tout a fait acceptable de 1,41%.

La mesure de courant a été réalisée avec la pince ampèremétrique. La plage de courant est d'après les caractéristiques: 0 à Icc=6.15A. La pince ampèremétrique sort 100mV pour 1A mesuré. On pourrait donc l'utiliser directement sur une entrée du PIC ou bien l'amplifier grâce à un AOP monté en amplificateur pour obtenir une meilleure précision. La mesure étant bruitée, il est aussi nécessaire de la filtrer avant l'amplification. Le schéma suivant a donc été choisi pour remplir ces deux exigences :



Le premier montage rempli donc la fonction filtre passe-bas / amplification et le deuxième est un montage suiveur pour ré-inverser le signal. La fonction de transfert du premier montage est la suivante :

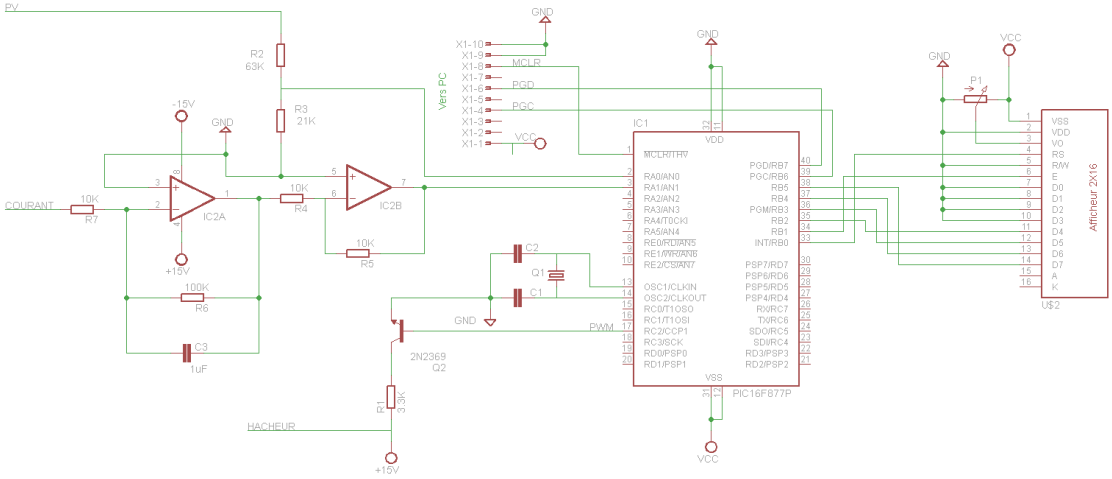
$$F(p) = -\frac{R_2}{R_1} \times \frac{1}{1 + R_1 C p}$$

Nous voyons apparaître dans le premier terme le facteur d'amplification et dans le deuxième un filtre du premier ordre avec une fréquence de coupure $f_c = 2\pi R_1 C$

La fréquence de coupure choisie est très basse pour éliminer le bruit et garder uniquement la partie continue du signal.

Les valeurs de composants choisies sont :
R1=10KΩ
C1=1uF
Ce qui donne une fréquence de coupure : $f_c = 15{,}9\ Hz$
Pour l'amplification, un facteur 10 permet d'obtenir une mesure de courant transposée sur 0 – 6V. Ce qui parait être un bon choix puisqu'on restera dans tous les cas en dessous de 5V à cause de la limitation de courant du hacheur. Pour satisfaire cette exigence il faut donc choisir R2=100KΩ.
Pour le deuxième montage en choisissant R3=R4=10KΩ on obtient un gain unitaire et donc la fonction d'inverseur.

Schéma complet :



IV.3 *Algorithme et code*

Le cahier des charges que je m'étais fixé pour l'algorithme était le suivant :

- ⇨ Génération de la MLI pour contrôler le rapport cyclique du hacheur en vue de l'obtention de la puissance maximale ⇔ faire varier le point de fonctionnement sur la courbe P(V) : Principe MPPT (Maximum Power Point Tracker)
- ⇨ Protection de la batterie en surcharge et en sous-charge et donc gestion de différents modes de charge ('trickle charge' lorsque la tension maxi est atteinte, commande de déconnection en dessous d'un seuil bas de tension)

Il faut donc définir 1 entrée du PIC pour la mesure de la tension de batterie et 2 entrées pour les mesures de tension et de courant du module PV.

1ère étape :MPPT

Pour éviter de prendre en compte des variations qui pourraient être dues à du bruit dans les mesures, le programme doit s'assurer de suivre une tendance. Dans ce but il doit donc garder en mémoire des mesures précédentes. J'utilise donc un tableau pour l'historisation des valeurs de puissances :

	Puissance	Sens
dernière valeur	[0]	[0]
	[1]	[1]
	…	…
	[n-1]	[n-1]
	[n]	[n]

Dans un premier temps j'ai cherché à faire fonctionner la MPPT en regardant simplement la variation sur la valeur courante et la précédente, j'utilise donc dans le code suivant un tableau de taille 2.

Code C:

```
unsigned int ValeurADTension;  //lecture de la tension PV
unsigned int ValeurADCourant;  //lecture du courant
unsigned int ValeurPuissance;  //puissance calculée
unsigned int duty_ratio;       //rapport cyclique
int LowLimit;                  //limite basse rapport cyclique
int HighLimit;                 //limite haute rapport cyclique
char Alpha[7];
char TENSION[7];
char COURANT[7];
char PUISSANCE[7];
int counter;                   //compteur gestion temps affichage
int DeltaAlpha;                //pas de variation du rapport cyclique
int Puissances[2];             //historique des puissances
int Sens[2];                   //historique du sens de variation du rapport cyclique
int i;

int VariationPuissance(int, int);   //Fonction : comparaison de 2 puissances

void main()
{
  //initalisation des butées
  LowLimit = 26; //10% de 255
  HighLimit = 230; //90% de 255

  PORTD = 0;            // Initialise PORTD
  TRISD = 0;            // Configure PORTD en sorties
  PORTC = 0;            // Initialise PORTC
  TRISC = 0;            // Configure PORTC en sorties
  TRISB = 0;            // Configure PORTB en sorties
  ADCON1 = 0x84;        // A0, A1, A3 entrées analogiques
  TRISA = 0xFF;         // Configure PORTA en entrées

   //initialisation port LCD
  Lcd_Config(&PORTB, 0,1,WR,5,4,3,2);
  Lcd_Cmd(Lcd_CURSOR_OFF);

  //initialisation de la fréquence de la PWM
  Pwm_Init(30000);   //30kHz
  //initialisation du rapport cyclique à 50%
  duty_ratio = 125;
  Pwm_Change_Duty(duty_ratio);

  //PWM start
  Pwm_Start();
```

```
//initialisation compteur pour affichage
counter=0;

//delta de variation du rapport cyclique
DeltaAlpha=50;

//intialisation tableaux de puissances / sens de variation du rapport cyclique
for (i=0;i<1;i++)
{
 Puissances[i]=0;
 Sens[i]=1;
}

while(1)
{
  Lcd_Cmd(Lcd_CLEAR);

  //lecture de la valeur de tension
  ValeurADTension = Adc_Read(0);
  IntToStr(((ValeurADTension*20)/1.024), TENSION); //ramené à 5V

  //lecture de la valeur de courant
  ValeurADCourant = Adc_Read(1);
  IntToStr(((ValeurADCourant*5)/1.024), COURANT);

  //Calcul de la puissance
  ValeurPuissance=((ValeurADTension)/102.4)*((ValeurADCourant)/102.4);
  IntToStr((ValeurPuissance), PUISSANCE);

  //mise à jour historique puissance
  Puissances[1]=Puissances[0];
  Puissances[0]=ValeurPuissance;

  //sens de variation du duty_cycle
  switch (VariationPuissance(Puissances[0],Puissances[1]))
  {
   case 1: // dernière valeur inférieure à la précédente
        Sens[0]=-Sens[1];
        break;
   case 2: // dernière valeur supérieure à la précédente
        Sens[0]=Sens[1];
        break;
  }

  //mise à jour tableau sens rapport cyclique
  Sens[1]=Sens[0];

  //variation du rapport cyclique
  duty_ratio = duty_ratio + (Sens[0]*DeltaAlpha);

  //vérification du dépassement des seuils
  if (duty_ratio < LowLimit)
  {
    duty_ratio = LowLimit;
  }
  if (duty_ratio > HighLimit)
  {
    duty_ratio = HighLimit;
  }
  IntToStr(duty_ratio,Alpha);
```

```
    //affichage
    counter++;
    if (counter<10)
    {
      Lcd_Out(1, 1, "COURANT=");
      Lcd_Out(1, 9, COURANT);
      Lcd_Out(2, 1, "TENSION=");
      Lcd_Out(2, 9, TENSION);
    }
    else
    {
      Lcd_Out(1, 1, "PUISSANCE=");
      Lcd_Out(1, 11, PUISSANCE);
      Lcd_Out(2, 1, "ALPHA=");
      Lcd_Out(2, 7, Alpha);
      if (counter==80)
      {
       counter=0;
      }
    }

    //envoi nouveau rapport cyclique
    Pwm_Change_Duty(duty_ratio);

    //clignotement LED
    PORTD=~PORTD;

    Delay_ms(50);
  }

}

/*************************************************************************/
/**********************************FONCTIONS**********************/
/*************************************************************************/

//comparaison des puissances
int VariationPuissance(int Val1, int Val2)
{
 if (Val1>Val2)
 {
   return 2;
 }
 else
 {
   return 1;
 }
}
```

Le programme a été testé et fonctionne bien avec un pas de variation de rapport cyclique important.

V améliorations possibles

Je n'ai hélas pas eu le temps de finir le projet tel que je l'avais défini dans mon cahier des charges. Il faudrait déjà garder en mémoire plus de valeurs de puissance et vérifier que la dernière valeur lue suit une tendance avant de la prendre en compte. Une solution rapide et simple serait de faire une moyenne des dernières valeurs de puissance pour limiter les effets de bruits.

Il serait aussi intéressant de faire varier le pas de variation en fonction de la variation de la puissance (valeur absolue de : mesure courante – dernière mesure).

En ce qui concerne la limite de charge, il faudrait d'abord regarder la tension de batterie au début de la boucle principale et rentrer dans une boucle de type « tant que » dans laquelle : soit on arrêtera la sortie MLI pour arrêter la charge, ou bien on essayera de suivre un courant de charge minimum tant que la tension de la batterie n'est pas retombée en dessous d'un seuil fixé.